EPJ Web of Conferences **245**, 01033 (2020) https://doi.org/10.1051/epjconf/202024501033
CHEP 2019

AliECS: a New Experiment Control System for the ALICE Experiment

Teo Mrnjavac[1,*], *Konstantinos* Alexopoulos[1,**], *Vasco* Chibante Barroso[1,***], and *George* Raduta[1,****]

[1]CERN, Geneva, Switzerland

Abstract. The ALICE Experiment at CERN's Large Hadron Collider (LHC) is undertaking a major upgrade during LHC Long Shutdown 2 in 2019-2021, which includes a new computing system called O^2 (Online-Offline). To ensure the efficient operation of the upgraded experiment and of its newly designed computing system, a reliable, high performance, and automated experiment control system is being developed. The ALICE Experiment Control System (AliECS) is a distributed system based on state of the art cluster management and microservices that have recently emerged in the distributed computing ecosystem. Such technologies will allow the ALICE collaboration to benefit from a vibrant and innovating open source community. This communication describes the AliECS architecture. It provides an in-depth overview of the system's components, features, and design elements, as well as its performance. It also reports on the experience with AliECS as part of ALICE Run 3 detector commissioning setups.

1 Introduction

1.1 The O^2 Computing System

The ALICE experiment [1] is undergoing a major upgrade [2] that is being deployed during the LHC's Long Shutdown 2 (2019-2021) in preparation for the LHC Run 3. The new and upgraded detectors will operate at a significantly increased data rate, and in order for the data processing to keep up, a new computing system called O^2 [3] is being designed, developed and deployed.

In its production stage, the O^2 computing system will consist of 100,000s processes, deployed over roughly 1000 heterogeneous nodes, fulfilling roles including data readout, processing, storage and auxiliary services. The system will read out 27 Tb/s of raw data and record 800 Gb/s of reconstructed data.

The O^2 computing system will be capable of two kinds of data-driven workflows: synchronous operation, intended to be *synchronous* with the detector readout, and asynchronous operation, which will take place at any time regardless of detector or beam conditions. Most

*e-mail: teo.m@cern.ch
**e-mail: kostas.alexopoulos@cern.ch
***e-mail: vmcb@cern.ch
****e-mail: george.raduta@cern.ch

nodes are expected to run dozens of processes of different kinds, including long running services, WLCG-like (Worldwide LHC Computing Grid) environments for asynchronous processing, and data-driven process workflows. Since synchronous workflows operate on data coming from detector data links, they must run in the O^2 facility at the LHC Point 2. Asynchronous workflows do not have this constraint, so they can run at any time on WLCG nodes, or on O^2 facility resources when they are not needed for synchronous operation.

1.2 The O^2/FLP Computing Cluster

The O^2 data processing workflows will run on two typologies of computing nodes: FLPs (First Level Processors) and EPNs (Event Processing Nodes). Each FLP is fitted with CRU (Common Readout Unit) [4] or C-RORC (Common Readout Receiver Card) [5] hardware, depending on the detector. These PCI-Express cards are capable of two way communication with detector front end electronics. Unlike FLPs, which host the first portion of the data flow, EPNs do not have physical links to detector hardware, and are instead configured as homogeneous computing nodes, operating as a second level of data processing after FLPs.

While O^2 is developed as a complete solution for the data processing needs of the ALICE experiment during Run 3, the O^2 computing system is split up in two separate computing clusters due to significant differences in requirements between FLPs and EPNs. This partition yields the O^2/FLP computing cluster and the O^2/EPN computing cluster, both deployed at the LHC Point 2.

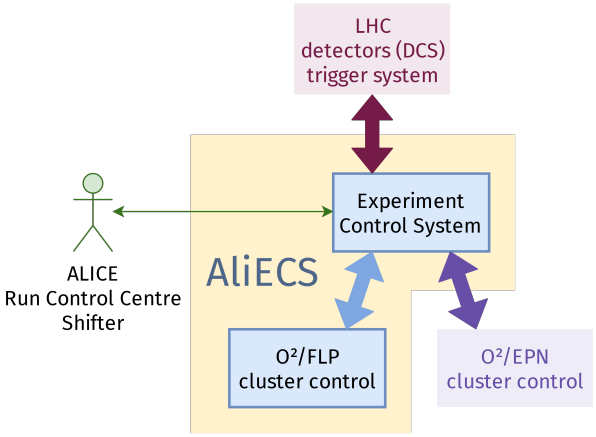

Figure 1. O^2/FLP and O^2/EPN cluster control with respect to the ALICE Run Control Centre.

The fundamental difference between these two kinds of nodes stems from the fact that FLPs have direct fiber links to detector front end electronics, making them permanently bound to a specific detector or detector component. Different FLPs may also have a variable number of CRU or C-RORC cards, and different system specifications. FLPs are not interchangeable, thus the O^2/FLP cluster is inevitably a heterogeneous environment. On the other hand, EPNs do not have direct links to detector front end electronics, and are largely interchangeable.

While the two computing clusters have their own specialised control mechanisms, the O^2 system as a whole will be controlled via a single user interface, an ECS (Experiment Control System) solution in the ALICE Run Control Centre (see Fig. 1).

1.3 Target Operational Improvements in an Experiment Control System for ALICE Run 3

The goals and requirements of the ALICE experiment control system (AliECS) are derived from experience in running the previous computing system during the LHC Runs 1 and 2 [6], and are motivated by a desire for greater reliability, performance, maintainability, and operational flexibility. Some specific operational goals refer to scenarios such as including or excluding a detector from data taking, during which it is desirable to reduce the time spent reconfiguring the data flow.

The O^2 project also includes a redesign of user interfaces, in favor of next-generation web-based GUIs with SSO (single sign-on) and a revamped look and feel. AliECS comes with command line and graphical user interfaces, including shifter oriented GUIs which supersede those of the previous generation ECS. Finally, the O^2 project is an opportunity to take advantage of modern developments in computing; AliECS is built with the best practices of a microservices distributed application paradigm, and harnessing the features of modern cluster resource management solutions.

2 Requirements of an ECS solution for ALICE Run 3

The primary duty of a control mechanism for the ALICE O^2 system is to launch, configure and control a set of data-driven workflows inside a computer cluster. On top of this cluster control role, AliECS is in charge of

1. managing the lifetime of thousands of processes in the O^2/FLP cluster (while delegating control of O^2/EPN processes to a specialized control mechanism for the O^2/EPN cluster),
2. minimizing the waste of beam time by reusing processes and avoiding time-consuming process restart operations,
3. and interfacing with the LHC, the trigger system, the DCS (Detector Control System) [7] and other systems through common APIs.

3 AliECS design overview

Due to the tight coupling required between high-level experiment control and O^2/FLP cluster control, AliECS integrates both experiment control and O^2/FLP cluster control into a single system. Thus, AliECS provides in-depth control of every data-driven process running in the O^2/FLP cluster. An interface between the AliECS core and the O^2/EPN control mechanism is under development in order to achieve coarse-grained, high-level control of the O^2/EPN cluster.

AliECS is a distributed system in charge of the O^2 facility with full knowledge and control over the resources of the O^2/FLP cluster. It implements a distributed state machine to represent the aggregated state of the constituent O^2 processes of a data-driven workflow. Furthermore, it allows reconfiguration and reuse of running O^2 processes as often as possible to avoid process restarts, and simultaneous operation of multiple workflows, with easy reallocation of resources among workflows. Finally, it reacts promptly to inputs, handling events from the user, the LHC, the trigger system, the DCS, and the cluster itself with a high degree of autonomy.

The O^2 project has chosen FairMQ [8] as the common message passing and data transport framework for its data-driven processes. It has been developed in the context of FairRoot [9, 10], a simulation, reconstruction and analysis framework for particle physics experiments. FairMQ provides the basic building blocks to implement complex data processing workflows, including a message queue, a configuration mechanism, a state machine, and a plugin system.

3.1 Resource Management in the O^2/FLP Facility

We implement AliECS as a distributed application, using Apache Mesos [11, 12] as toolkit. This custom solution integrates a task scheduler component, a purpose-built distributed state machine system, a multi-source stateful process configuration mechanism, and a control plugin and library compatible with any data-driven O^2 process.

Apache Mesos is a cluster resource management system, which facilitates the management of O^2/FLP components, resources and tasks inside the O^2/FLP facility, effectively enabling the developer to program against the datacenter (i.e., the O^2/FLP facility at LHC Point 2) as if it was a single pool of resources. Mesos is an open source project hosted by the Apache Software Foundation, and used in deployments of 10,000s nodes.

For AliECS, benefits of using Mesos include knowledge of what runs where, resource management (including port assignment), transport facilities for O^2-specific control messages, task status tracking (e.g. an event is raised if a task dies unexpectedly), as well as advanced features such as node attributes, resource overprovisioning, checkpointing, and others.

3.2 AliECS Components

AliECS is under development as our solution for the problem of O^2/FLP synchronous control and ECS. The current implementation of AliECS (see Fig. 2) can be found on GitHub [13], and it consists of

1. the AliECS core (which includes the Apache Mesos-facing scheduler component),
2. the AliECS executor,
3. the O^2 control and configuration plugin for FairMQ devices (OCC plugin),
4. the O^2 control and configuration library (OCC library),
5. the AliECS control and configuration command line utility (`coconut`),
6. the AliECS process execution and control utility for OCC library based O^2 processes (`peanut`),
7. and the web-based AliECS GUI.

AliECS interfaces with Consul [14], a key-value store that acts as the system's configuration repository. The design also includes interfacing with information sources from the LHC, the trigger system and the DCS. Once acquired by the AliECS core, configuration information is processed into an in-memory hierarchical key-value store, and then it is fed into a template system in order to generate task deployment and configuration structures.

Most components of AliECS are written in Go [15], a statically typed general purpose programming language in the tradition of C, which is particularly suitable for distributed system development because of its advanced synchronization and threading facilities. The OCC plugin is developed in C++17 and works with any FairMQ-based process. A non-plugin library equivalent of the latter is also provided, for O^2 processes which do not support the FairMQ plugin system. The common idiom of inter-process communication in AliECS is gRPC [16], an open source, cross-language RPC (Remote Procedure Call) system that is widely used in the microservices community.

3.3 AliECS Concepts

The basic unit of scheduling in AliECS is a *task*. A task generally corresponds to a process, more specifically a process that can receive and respond to OCC-compatible control messages. All AliECS workflows are collections of tasks, which together form a coherent data processing chain.

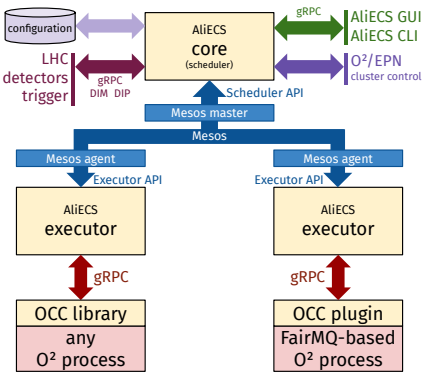

Figure 2. The AliECS architecture. All control communications between core and executor instances are piggybacked on Mesos messages. The OCC plugin hides the complexities of driving the state machine of the controlled process.

Tasks are the leaves in a tree of *roles*. A role is a runtime subdivision of the complete system, it represents a kind of operation along with its resources. Roles allow binding tasks or groups of tasks to specific host attributes, detectors and configuration values. In memory, a tree of O^2 roles, along with their tasks and their configuration is a *workflow*. A workflow aggregates the collective state of its constituent O^2 roles. A running workflow, along with associated detectors and other hardware and software resources required for experiment operation constitutes an *environment*, which is the highest level of state machine control (see Fig. 3).

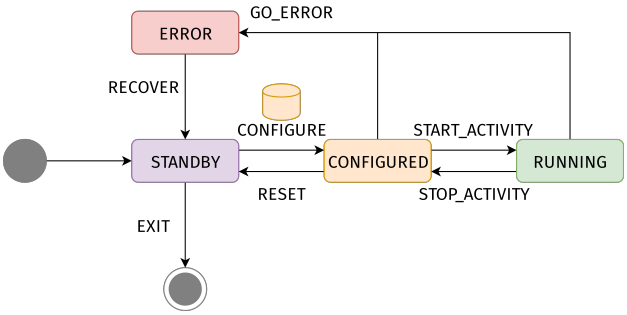

Figure 3. The state machine of an AliECS environment. The same state machine is implemented by each task. For FairMQ-based tasks the OCC plugin acts as a translation layer between the AliECS task state machine and the underlying FairMQ state machine.

3.4 Configuration Management

AliECS is both a producer and consumer of configuration data in the O^2/FLP cluster. There are three kinds of configuration information that AliECS deals with:

1. the AliECS core configuration,
2. the AliECS workflow configuration,
3. and the O^2 tasks configuration.

The AliECS core configuration is a flat list of read-only values that the AliECS core acquires on startup. Typical values that come from this configuration mechanism are the control port to use for incoming AliECS GUI or `coconut` connections and the URI of the Mesos master API.

The AliECS workflow configuration is acquired by way of a configuration manager subsystem that uses Git repositories as a backend for file storage and versioning. The AliECS workflow configuration data consist of task descriptor files and workflow template files, sourced from Git repositories. A task descriptor file is a YAML document that describes how to launch and control a single task, such as an O^2 data-driven process. A workflow template file is a YAML document that describes the structure of a workflow of roles and (ultimately) tasks. This structure directly expresses the control tree, which defines the layout of the distributed state machine.

Workflow configuration is further complemented by AliECS runtime variables, which can affect the loaded workflow and single tasks.

AliECS implements task configuration as a push operation associated with the `CONFIGURE` transition, the payload of which includes communication channel configuration (i.e. hosts and ports to connect or bind) as well as an optional key-value map of application-specific configuration data.

3.5 O^2 Process Control

Most O^2 processes are also FairMQ devices, i.e., programs that make use of the FairMQ library for its state machine and I/O facilities.

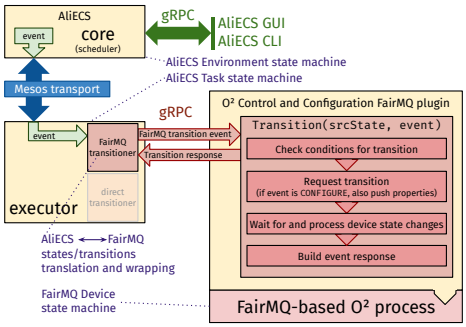

Figure 4. The AliECS executor integrates modular components called *transitioners* which act as translation wrappers for the state machine of a specific controlled process. In the figure above the executor has loaded the FairMQ transitioner, which drives the state machine of a FairMQ-based process.

FairMQ provides a plugin system that is capable of loading the purpose-built O^2 control and configuration plugin for FairMQ devices. This plugin enables any FairMQ device to accept control commands from an AliECS executor (see Fig. 4). When the OCC plugin receives a remote procedure call from the executor, it drives the state machine of the FairMQ device and reports back. The OCC plugin is also capable of pushing configuration key-value pairs as FairMQ properties to the FairMQ configuration map of the device.

4 AliECS in Run 3 detector commissioning

An Ansible-based [17] installation system for AliECS and other O^2 components is being developed in order to facilitate the deployment in ALICE Run 3 detector commissioning setups as well as in the O^2/FLP cluster. This installation mechanism can be used via Foreman [18], a server lifecycle management solution, or through a custom command line installer tool, available for both single-node and multi-node deployments.

AliECS instances for detector commissioning tasks include setups for the ALICE Time Projection Chamber [19], Inner Tracking System [20] and Muon Forward Tracker [21] subdetectors. As we performed these deployments we encountered the following major challenges:

1. AliECS is often deployed in an environment where ALICE detector teams already have their own tooling and scripts, which complicates integration.
2. The IPC interfaces between AliECS and the DCS, and between AliECS and the trigger system aren't in place yet, which requires workarounds.
3. AliECS instances for detector teams need to be deployed either off-premises, or on-premises but in differently configured network environments, which complicates automation and support intervention.
4. AliECS integrates with a multitude of O^2 components, which makes integration testing critical for successful releases.

We have promptly reacted to these challenges by collecting further requirements from detector teams, and by more clearly communicating the potential integration points between detector team tooling and AliECS components. We have extended and improved `coconut`, which can easily be called within shell scripts to direct AliECS behavior, and we have engaged to extend AliECS so it can also execute generic commands, as opposed to only stateful OCC-compatible tasks.

We have further extended and refined our Foreman-based system configuration management facilities, and we have developed a new Ansible-based multi-node installer system written in Go, as a replacement for the previously used wrapper script. A high level testing mechanism for the Ansible roles which install AliECS and other O^2 components was also developed, in order to spot integration issues as early as possible.

5 Conclusion

We propose a new, custom built, microservices oriented, integrated solution for ALICE experiment control as well as for cluster control in the FLP facility of the O^2 computing system. We assert that the leap to O^2 is an opportunity for a broad technical refresh by leveraging modern cluster resource management and IPC technologies for a high performance, low latency ECS.

By taking advantage of Apache Mesos, we gain resource management, control message transport, events, and more, with the goal of achieving improved operational flexibility. On top of this framework, we implement a distributed state machine mechanism, with an expressive configuration format and a modular process control stack for maximum compatibility in an inevitably heterogeneous context. We employ open source cross-platform and cross-language technologies such as gRPC, Git and Consul to maximize interoperability and minimize technical risks.

We aim to maximize the usage of LHC beam time while ensuring optimal resource allocation in the new O^2 facility for both synchronous and asynchronous data-driven workflows. AliECS takes direct control over the O^2/FLP facility, and interfaces with the O^2/EPN cluster control to gain high-level oversight of the whole data readout chain. With our design approach we aim to achieve substantial performance improvements and operational benefits in mission critical use cases compared to the previous system.

References

[1] K. Aamodt et al. (ALICE), JINST **3.08**, S08002 (2008)
[2] B. Abelev et al. (ALICE), Journal of Physics G: Nuclear and Particle Physics **41**, 087001 (2014)
[3] J. Adam et al. (ALICE), Tech. Rep. CERN-LHCC-2015-006 / ALICE-TDR-019, CERN (2015)

[4] J. Mitra, S. Khan, S. Mukherjee, R. Paul, Journal of Instrumentation **11**, C03021 (2016)
[5] A. Borga, F. Costa, G. Crone, H. Engel, D. Eschweiler, D. Francis, B. Green, M. Joos, U. Kebschull, T. Kiss et al., Journal of Instrumentation **10**, C02022 (2015)
[6] F. Carena, W. Carena, S. Chapeland, V.C. Barroso, F. Costa, E. Dénes, R. Divià, U. Fuchs, A. Grigore, T. Kiss et al., Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **741**, 130 (2014)
[7] P. Chochula et al., Proceedings of the 16th International Conference on Accelerator and Large Experimental Control Systems pp. 323–327 (2018)
[8] *FairMQ C++ Message Queuing Library and Framework*, https://github.com/FairRootGroup/FairMQ, accessed: 2019-09-26
[9] M. Al-Turany, D. Bertini, R. Karabowicz, D. Kresan, P. Malzacher, T. Stockmanns, F. Uhlig, Journal of Physics: Conference Series **396**, 022001 (2012)
[10] M. Al-Turany, P. Buncic, P. Hristov, T. Kollegger, C. Kouzinopoulos, A. Lebedev, V. Lindenstruth, A. Manafov, M. Richter, A. Rybalchenko et al., Journal of Physics: Conference Series **664**, 072001 (2015)
[11] *Apache Mesos*, http://mesos.apache.org/, accessed: 2019-09-26
[12] D. Berzano, G. Eulisse, C. Grigoraş, K. Napoli, Journal of Physics: Conference Series **898**, 082043 (2017)
[13] *The O^2 Control System*, https://github.com/AliceO2Group/Control, accessed: 2019-09-26
[14] *Consul by HashiCorp*, https://www.consul.io/, accessed: 2019-09-26
[15] *The Go Programming Language*, https://golang.org/, accessed: 2019-09-26
[16] *gRPC A high performance, open-source universal RPC framework*, https://grpc.io/, accessed: 2019-09-26
[17] *Red Hat Ansible*, https://www.ansible.com/, accessed: 2019-09-19
[18] *Foreman*, https://theforeman.org/, accessed: 2019-09-19
[19] J. Adam et al. (ALICE), Tech. Rep. CERN-LHCC-2013-020 / ALICE-TDR-016, CERN (2013)
[20] B. Abelev et al. (ALICE), Tech. Rep. CERN-LHCC-2013-024 / ALICE-TDR-017, CERN (2014)
[21] J. Adam et al. (ALICE), Tech. Rep. CERN-LHCC-2015-001 / ALICE-TDR-018, CERN (2015)